DOCUMENT RESUME

ED 107 298 IR 002 048

AUTHOR	Brown, John Seely; And Others
TITLE	SOPHIE: A Sophisticated Instructional Environment. Final Report for Period January 1974 through June 1974.
INSTITUTION	Air Force Human Resources Lab., Brooks AFB, Texas.
REPORT NO	AFHRL-TR-74-93
PUB DATE	Dec 74
NOTE	51p.
EDRS PRICE	MF-$0.76 HC-$3.32 PLUS POSTAGE
DESCRIPTORS	*Computer Assisted Instruction; Educational Development; *Educational Environment; Educational Improvement; Educational Innovation; Educational Research; *Electromechanical Aids; Instructional Materials; Instructional Media; *Instructional Systems; Instructional Technology; Learning Laboratories; Military Schools; Programed Instruction; Programing Languages; *Simulation; Teaching Procedures
IDENTIFIERS	ARPA Network; *SOPHIE; Sophisticated Instructional Environment

ABSTRACT

The SOPHIE program, which implements mixed initiative computer-assisted instruction within a simulated electronics trouble shooting training laboratory interaction, has been extended in several ways. The language processor now accepts ellipses and other nonspecific requests and resolves these from dialog context. A help requesting facility has been provided which will suggest possible faults (based on the student's knowledge about the circuit at the time of request) which could explain the symptoms he has observed. The net effect of modifications is that a dialog is much more like a conversation with a very skilled tutor who can infer what a students means, based on a complete interaction session, and respond appropriately. The resulting program can be accessed through the ARPA network of computers. (SK)

ED107298

AFHRL-TR-74-93

AIR FORCE HUMAN RESOURCES LABORATORY

SOPHIE:
A SOPHISTICATED INSTRUCTIONAL ENVIRONMENT

By

John Seely Brown
Richard R. Burton
Alan G. Bell
Robert J. Bobrow
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

TECHNICAL TRAINING DIVISION
Lowry Air Force Base, Colorado 80230

December 1974
Final Report for Period January 1974 – June 1974

Approved for public release; distribution unlimited.

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

IR 000048

NOTICE

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This final report was submitted by Bolt Beranek and Newman Inc., 50 Moulton Street, Cambridge, Massachusetts 02138, under contract F41609-74-C-0015, project 1121, with Technical Training Division, Air Force Human Resources Laboratory (AFSC), Lowry Air Force Base, Colorado 80230. Mr. Edward M. Gardner was the contract monitor.

This report has been reviewed and cleared for open publication and/or public release by the appropriate Office of Information (OI) in accordance with AFR 190-17 and DoDD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved.

MARTY R. ROCKWAY, Technical Director
Technical Training Division

Approved for publication.

HAROLD E. FISCHER, Colonel, USAF
Commander

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFHRL-TR-74-93

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle):
SOPHIE:
A SOPHISTICATED INSTRUCTIONAL ENVIRONMENT

5. TYPE OF REPORT & PERIOD COVERED:
Final
January 1974 - June 1974

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s):
John Seely Brown
Richard R. Burton
Alan G. Bell
Robert J. Bobrow

8. CONTRACT OR GRANT NUMBER(s):
F41609-74-C-0015

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
62703F
11210205

11. CONTROLLING OFFICE NAME AND ADDRESS:
Hq Air Force Human Resources Laboratory (AFSC)
Brooks Air Force Base, Texas 78235

12. REPORT DATE:
December 1974

13. NUMBER OF PAGES:
48

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
Technical Training Division
Air Force Human Resources Laboratory
Lowry Air Force Base, Colorado 80230

15. SECURITY CLASS. (of this report):
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):
computer assisted instruction
language processor
computer dialogue
instructional environment
technical training
parsing
semantic information

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):
The SOPHIE program, which implements mixed initiative CAI within a simulated electronics troubleshooting training laboratory interaction, has been extended in several manners. The language processor now accepts ellipses and other nonspecific requests and resolves these from dialogue context. A help requesting facility has been provided which will suggest possible faults (based on the student's knowledge about the circuit at the time of request) which could explain the symptoms he has observed. The net effect of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT

modifications is that a dialogue is much more like a conversation with a very skilled tutor who can infer what a student means, based on a complete interaction session, and respond appropriately. The resulting program can be accessed through the ARPA network of computers.

SUMMARY

Objective

The SOPHIE program has now been extended to the point where a person capable of troubleshooting an electronic device and capable of talking with a human tutor should be able to carry out a meaningful dialogue with it. The modifications performed on its grammar processing allow for context dependent impressive specification of requests such as occur in a casual conversation between people working on devices in an electronics repair shop. The added ability to request suggestions on what might be investigated next corresponds closely to a request which a student might expect to make of a human instructor. The instructor looking at what the student sees as symptoms would direct him to several general possibilities and expect the student to isolate those possibilities.

Approach

SOPHIE has reached a point where further development requires feedback from student usage to be effective. This should be considered in any further development effort. Feedback has thus far been obtained from people with background and skills exceeding that of the target trainee and is thus only partially relevant as an evaluative situation.

Recommendation

Not yet investigated is the potential for use of a SOPHIE like program by instructional developers as an authoring aid. It is clear now that programmed text on CAI material on the IP-28 power supply could be very easily authored using SOPHIE as a communicative expert on how this circuit operates or even more directly as the producer of a data base from which simpler paradigms than the mixed-initiative could be devised. In this sense SOPHIE should be thought of as the beginning of a new approach to authoring as well as a sophisticated form of instruction. The possibilities of using SOPHIE type techniques in the training of such areas as aircraft flight should also not be ignored. The ramifications of an "intelligent" programmed expert carefully "watching" a student fly a simulator and telling him in detail where he failed in real time might prove to be very economically viable.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 3
SOPHIE's Goals and Objectives 4
Reasons for Choosing Electronic Troubleshooting as SOPHIE's First Domain of Expertise 5
Basic Scenario 6
Annotated Dialogue 7
Techniques for Inferencing 14
Hypothesis Evaluation 15
Simulation Models 17

CHAPTER 2: HYPOTHESIS GENERATION 18
Interaction of the Three Specialists 19
The Proposer 19
The Instantiator 20
The Refiner 22
Measurement Verification 23

CHAPTER 3: A SEMANTICALLY CENTERED PARSING SYSTEM 24
Annotated Dialogue 25
Use of Semantic Information During Parsing 29
Semantic Prediction in the Grammar 29
Using Local Information for Simple Omissions 30
Recognition of Ellipses 31
Capturing Semantic Information in the Parser 32
Representation of Information in the Grammar 33
Result of the Parsing 33
Determining Pronoun Referents 34
Determining Elliptic Referents 34
Fuzziness 35
Prescanning 35
Conclusions 37

REFERENCES 38

APPENDIX A: FORMAL DESCRIPTION OF PART OF THE GRAMMAR 40

APPENDIX B: A RULE FROM THE GRAMMAR 42

APPENDIX C: SAMPLE PARSES AND PARSE TIMES 45

CHAPTER 1 -- INTRODUCTION

This report covers our research and development efforts on SOPHIE* for the six month period ending June 30, 1974. The first part of the report reviews the overall goals of the SOPHIE project and then presents an annotated dialogue of a student using the most recently released version of the system. This dialogue exhibits many of the features that have been developed during this six month period. Following the dialogue we include a brief but self-contained description of the basic inferencing techniques which enable this system to achieve its question answering, hypothesis evaluation, and hypothesis generation behavior. It is intended that this first chapter contain sufficient detail that the reader need not have studied our prior final report in order to understand the remaining chapters.

The second and third chapters of the report concentrate on the new features of SOPHIE. Chapter 2 provides a detailed description of the theory formation or hypothesis generation abilities of this new version. Although the original system performed some limited hypothesis generation, we have greatly expanded this module and, more important, we have found a novel and powerful use for these extended capabilities. In the original version, hypothesis generation was used soley to provide a student with help when he ran out of viable ideas about what could be wrong with the instrument. During the last six months we have discovered that by making the hypothesis generation system "complete" (i.e. in the sense of it being able to construct all single fault theories that are logically consistent with the known measurements) we can use it to verify whether or not a new measurement is logically redundant with respect to the current or known set of measurements. In other words we can use this module to determine whether or not the next given measurement could in any way add new information about what could be wrong by checking to see if it reduces the list of possible faults which the hypothesis generation system produces! This, for example, would enable us to automatically grade a student's sequence of measurements.

Chapter 3 describes the major additions made to the natural language front-end of SOPHIE. By extending our use of a semantic grammar, we can now handle utterances that are incomplete sentences or that involve the use of pronoun references, etc. This new processor achieves this capability by using the previous questions (i.e., those just

---

*A SOPHisticated Instructional Environment

asked by the student) to establish a context for the dialogue. This context is then used to make explicit any of the implicit information in the student's current question. We think that this new natural language processor makes SOPHIE one of the first sytems that has successfully coped with the unique problems of man-machine dialogue and as such it makes it one of the most "habitable" or friendliest systems around.

SOPHIE's Goals and Objectives

SOPHIE represents a major step toward the goal of producing a "reactive" learning environment. In an ideal reactive environment the student is encouraged to explore ideas, create conjectures or hypotheses about a situation and then to receive immediate detailed feedback as to the logical validity of these ideas. In those cases where his ideas or proposed solutions have logical flaws, the system creates relevant counter-examples or critiques so that the student can start to debug his ideas. In short, a reactive learning environment extends Carbonell's original concept of mixed-initiative Computer Assisted Instruction (CAI) to the point where the student has a one-to-one relationship with an "expert" (system) which in some ways can surpass the inferential capabilities of most human tutors. Of course, creating a system that has both the depth and breadth of a human tutor is far beyond the current state of the art, but by carefully choosing a domain of knowledge for which we have extremely powerful inferencing mechanisms, we can create an artificially intelligent "expert" system which can patiently provide the student with a logically deep sounding board for his own ideas.

SOPHIE was designed to fulfill three main objectives: The first was to demonstrate that the notion of using Artificial Intelligence (AI) techniques to build an "intelligent" CAI system (ICAI) was not purely a pipe dream but that in fact a system could be built that was sufficiently complete and efficient that it could be used as an experimental tool in a classroom environment. The second objective was to explore some new dimensions for CAI which exploit the significant increase in computational power provided by current advances in hardware technology. It seemed fruitful to begin such an investigation today so that we are prepared to imaginatively utilize tomorrow's computers. The third was to fulfill the need for an environment in which to experiment with new ways of teaching problem solving skills, such as electronic troubleshooting, without being constrained to pose only problems having extensionally defined solution sets. We wanted to allow the

student freedom in choosing the way in which he could go about solving his problem while still expecting our system to monitor all his decisions and provide him with useful feedback without our having to anticipate (and hence program in) his every move, query etc.

The idea of using AI techniques in CAI was originated with Carbonell in his mixed-initiative SCHOLAR systems (Carbonell 1970, 1973). Since then, other systems for teaching symbolic logic (Goldberg 1973), meteorology (Brown et al., 1973) and the interpretation of nuclear magnetic resonance spectra (Sleeman 1974) have explored ways in which to augment the mixed-initiative system with considerably more problem solving and inferencing capabilities. Admittedly, much of the increased logical capabilities of these latter systems has been achieved at the cost of restricting the kinds of generic knowledge to be represented. However, this trade-off seems eminently reasonable since these latter systems are not trying to mimic all the roles of a human tutor.

SOPHIE reflects a major research effort to produce a CAI system that, on the one hand, produces deep logical inferences on a domain less formal than symbolic logic and, on the other hand, is sufficiently complete that it can answer nearly all questions posed to it by a student. To the extent that SOPHIE accomplishes this, it overcomes a major limitation inherent in nearly all intelligent systems. However, these capabilities are by their very nature complex and as such require a sophisticated set of strategies and procedures. For example, this kernel version of SOPHIE represents approximately 300,000 words (36 bit) of INTERLISP and FORTRAN code running on a virtual memory TENEX. Although it is an immense program, it is surprisingly efficient exhibiting a typical response delay of around three seconds on a lightly loaded system and requiring on the average about two cpu minutes per hour of student use.

Reasons for Choosing Electronic Troubleshooting as SOPHIE's First Domain of Expertise

There are several reasons that influenced our choice of electronic troubleshooting as the subject domain around which to build this system. The first is that it provides an excellent domain for developing and experimenting with a reactive learning environment. For example with the use of a simulator a student can experiment with a circuit by modifying its various components and examining the consequences of these modifications. Within the simulation context he can quickly make all kinds of measurements (some

of which would ordinarily require the time-consuming operation of decoupling a component from the circuit). He need never worry about limiting his experimentation through fear of blowing up the instrument. Indeed if this happens, the student can be directly informed that his last experiment blew certain components, or he could be told that something blew and be asked to troubleshoot his own mis-doings.

The second, and by far the most important reason for choosing this domain, is that the lab instructor seldom has the time to answer the individual questions which arise while the student is troubleshooting. Also, the instructor doesn't usually have the time to have each student articulate the train of hypotheses that he is developing while troubleshooting. Consequently, the instructor misses a crucial opportunity for providing the student with detailed logical analyses of the correctness of his hypotheses just when the student is most likely to be interested in such feedback. The reactive environment provided by SOPHIE has sufficient inferencing capabilities to circumvent all these limitations. (Note that such inferencing, or deductive capabilities, represent far more potential than just the obvious use of a simulator as first mentioned.)

Basic Scenario

The basic scenario underlying SOPHIE concerns a student attempting to isolate a fault in a given piece of electronic equipment while having a lab instructor standing over his shoulder to answer questions, evaluate his hypotheses and pose alternatives to him if he becomes stuck.

In the usual setting, SOPHIE presents the student with a circuit schematic of the instrument under study and automatically selects and inserts a fault of some specified degree of difficulty. The student then tries to debug the instrument by requesting various measurements. At any time, he can offer a hypothesis about what he thinks could be wrong with the instrument and receive an evaluation of his hypothesis, which specifies whether it is consistent with what he should have been able to infer from his measurements. The student can also, at any time, replace any component, but before a part is replaced, the student is queried as to what he thinks is wrong with it. If his answers are correct the component is replaced. In those cases where he has discovered a part which was blown because of a deeper fault, the replaced component will be reblown until he discovers and fixes the fundamental fault. If the

student becomes stuck and cannot think of any faults which would explain his measurements, he can ask for help. SOPHIE then examines his measurements and generates plausible hypotheses (with respect to all the previous measurements) which he can explore.

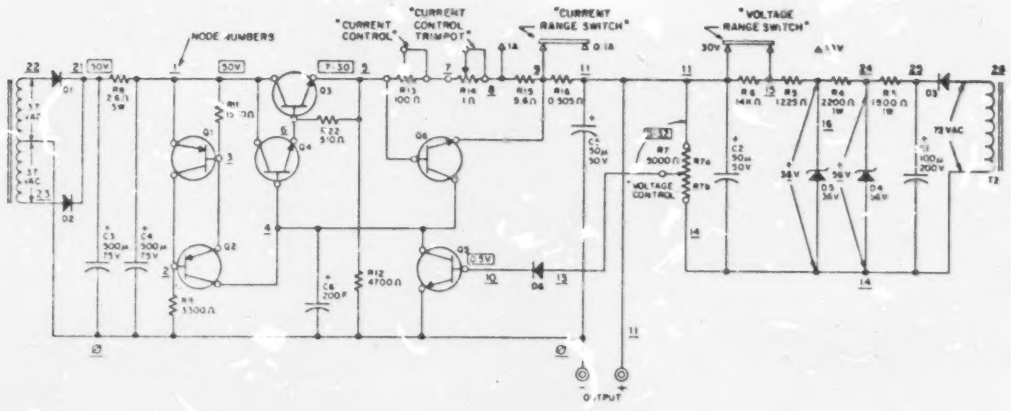
In order to illustrate some of SOPHIE's capabilities, we present a protocol generated by a student using SOPHIE. In this session the student takes some measurements, develops an idea, is told why that idea is wrong, gets stuck, is given some new ideas and eventually solves his problem.

Annotated Dialogue

The instrument currently being modeled is the Heathkit IP-28 regulated power supply which is a reasonably sophisticated, six transistor power supply with both current and voltage limiting behavior. As such it involves two critical feedback paths which make troubleshooting this instrument fairly complex.

PROTOCOL

WELCOME TO SOPHIE, A REACTIVE LEARNING ENVIRONMENT.
THE PIECE OF EQUIPMENT YOU ARE STUDYING IS THE IP-28 REGULATED POWER SUPPLY. WOULD YOU LIKE A CIRCUIT SCHEMATIC?*
> YES **



---

*The schematic has been photo-reduced to save space and is included here to give an idea of the complexity of the circuit.
**Parts typed by the student are underlined.

```
WOULD YOU LIKE TO START WITH A FAULTED INSTRUMENT WHICH NEEDS FIXING?
> YES
THE INSTRUMENT HAS A FAULT AND ITS PRESENT CONTROL SETTINGS ARE:
CC        1.0       (CURRENT CONTROL-FROM 0.0 (MIN CURRENT) TO 1.0)
CR        HIGH      (CURRENT RANGE SWITCH, HIGH=1 AMP, LOW=.1 AMP)
LOAD      20        (LOAD RESISTANCE IN OHMS)
VC        1.0       (VOLTAGE CONTROL-FROM 0.0 (MIN VOLTAGE) TO 1.0)
VR        HIGH      (VOLTAGE RANGE SWITCH, HIGH=30 VOLTS, LOW=10 VOLTS)
```

It is assumed that the student has already had an introduction to the basic principles of the power supply so the above comments are sufficient to characterize the controls.

```
>>WHAT IS THE OUTPUT?
THE OUTPUT VOLTAGE IS 11.7 VOLTS.

>>WHAT IS IT IN A WORKING INSTRUMENT?
IN A WORKING CIRCUIT THE OUTPUT VOLTAGE IS 19.9 VOLTS.
THAT VALUE IS INCORRECT.
```

In case the student doesn't know what the value of a measurement should have been, he can have any measurement repeated in a working instrument. Note the student's use of the pronoun "it" to refer to the entire previous sentence.

```
>>WHAT IS THE VOLTAGE BETWEEN R8 AND Q1
THE VOLTAGE AT N1 IS 47.6 VOLTS.

>>WHAT IS THE BSE EMITTER VOLTAGE OF THE VOLTAGE LIMITING TRANSITOR?
=BASE
=TRANSISTOR
THE BASE EMITTER VOLTAGE OF Q5 IS -.037 MILLIVOLTS.
```

The spelling corrector has detected and corrected two misspellings. The system must also know that Q5 is the voltage limiting transistor.

```
>>WHAT ABOUT FOR THE CURRENT LIMITING TRANSISTOR
THE BASE EMITTER VOLTAGE OF Q6 IS .349 VOLTS.
```

The meaning of this question is completely dependent on the context established by the prior sentence. The system uses this context to determine that he is interested in not only voltage measurements but the voltage across a particular junction.

```
>>WHAT SHOULD IT BE?
IN A WORKING CIRCUIT THE BASE EMITTER VOLTAGE OF Q6 IS .593 VOLTS.
```

Another way of requesting the prior measurement to be repeated

in a working instrument, i.e., the meaning of "should" implies that a working instrument should be used.

>>REPLACE Q6
I AM GOING TO ASK YOU SOME QUESTIONS ABOUT HOW Q6 IS FAULTED.
ARE ANY JUNCTIONS SHORTED?

Before the system replaces a component it queries him as to what he thinks is wrong. If ihe gets at least something right, the system tells him what was actually wrong and goes ahead and replaces it.

> I DON'T KNOW
ARE ANY TERMINALS OPEN?
> YES
WHICH TERMINALS ARE OPEN?
> THE BASE
Q6 IS NOT FAULTED IN THAT MANNER. Q6 NOT REPLACED.

>>IS IT POSSIBLE THAT Q6 HAS AN OPEN BASE

The student now has an idea what might be wrong. However before the system's evaluation procedures can be applied to his idea or hypothesis the hypothesis must be completely specified. To help the student out, the system suggests a plausible complete specification of it.

IS THE COMPLETE FAULT MODE EB/EC/BC = OP/OK/OP WHAT YOU INTENDED?
> SURE
WHEN THE SETTINGS WERE: CC=1.0, CR=HIGH, LOAD=20, VC=1.0, VR=HIGH
THAT FAULT CONFLICTS WITH THE FOLLOWING MEASUREMENT.
YOU OBSERVED THAT THE OUTPUT VOLTAGE WAS 11.7 VOLTS.
IF Q6 HAD FAULT MODE EB/EC/BC = OP/OK/OP IT WOULD BE 23.9 VOLTS.
IN A WORKING CIRCUIT IT IS 19.9 VOLTS.

In determining the logical consistency of his hypothesis with respect to the measurements he has already made, a glaring counter-example is discovered.

ALSO:
THAT FAULT CONFLICTS WITH THE FOLLOWING MEASUREMENT.
YOU OBSERVED THAT THE BASE EMITTER VOLTAGE OF Q6 WAS .349 VOLTS.
IF Q6 HAD FAULT MODE EB/EC/BC = OP/OK/OP IT WOULD BE .712 VOLTS.
IN A WORKING CIRCUIT IT IS .593 VOLTS.

The system then points out a more subtle counter-example.

THAT FAULT DOES NOT EXPLAIN ANY OF THE SYMPTOMS YOU OBSERVED BUT IT IS CONSISTENT WITH THE VALUES FOR
THE BASE EMITTER VOLTAGE OF Q5 AND THE VOLTAGE AT N1.

And finally the system tells him what is right or at least consistent with his hypothesis.

```
>>WHAT IS THE OUTPUT OF THE CONSTANT CURRENT SOURCE
THE OUTPUT OF THE CONSTANT CURRENT SOURCE IS -.59 MILLIAMPS.
```

To understand this request the system must also know about the functional blocks of the instrument. Also note that the system had to determine that "output" in this context meant "output current" as opposed to "output voltage" as was meant in the first questions of this protocol!

```
>>WHAT SHOULD IT BE?
IN A WORKING CIRCUIT THE OUTPUT CURRENT OF THE CONSTANT CURRENT SOURCE
 IS -.6 MILLIAMPS.

>>V BETWEEN N4 AND N5
THE VOLTAGE BETWEEN N4 AND N5 IS 1.64 VOLTS.
```

Noun phrase utterances get interpreted as questions.

```
>>COULD THE PROBLEM BE THAT THE BASE OF Q3 IS OPEN?
IS THE COMPLETE FAULT MODE EB/EC/BC = OP/OK/OP WHAT YOU INTENDED?
> SURE

HMMM, LET ME THINK ABOUT THAT FOR A MINUTE.
UNDER THE SETTINGS:
       CC=1.0, CR=HIGH, LOAD=20, VC=1.0, VR=HIGH
THE FAULT WHICH YOU PROPOSED WILL PROPAGATE.
LIKELY FAULTS WHICH MIGHT OCCUR ARE:
R22 OPEN

WOULD YOU LIKE ME TO CONTINUE TESTING YOUR HYPOTHESIS IN A CIRCUIT
IN WHICH R22 IS OPEN ALSO?
> NO
```

The student proposes a hypothesis that logically entails another component being blown. Before further evaluation of his hypothesis, the user is querried as to whether he realizes this entailment. If he doesn't, he is given the chance to have his hypothesis evaluated in a hypothetical or "ghost" world in which that component would not have blown (e.g. in this case the power rating of R22 would have to be increased to a value greater than its power dissipation.)

```
WOULD YOU LIKE ME TO COMMENT ON YOUR HYPOTHESIS AS IF NO OTHER PARTS
HAD BLOWN?
> YES
WHEN THE SETTINGS WERE:
       CC=1.0, CR=HIGH, LOAD=20, VC=1.0, VR=HIGH,
```

```
THAT FAULT WOULD HAVE CAUSED THE FOLLOWING SYMPTOM.
YOU OBSERVED THAT THE VOLTAGE BETWEEN N4 AND N5 WAS 1.64 VOLTS.
IF Q3 HAD FAULT MODE EB/EC/BC = OP/OK/OP IT WOULD BE 31.3 VOLTS.
IN A WORKING CIRCUIT IT IS 19.9 VOLTS.

ALSO:
THAT FAULT CONFLICTS WITH THE FOLLOWING MEASUREMENT.
YOU OBSERVED THAT THE OUTPUT VOLTAGE WAS 11.7 VOLTS.
IF Q3 HAD FAULT MODE EB/EC/BC = OP/OK/OP IT WOULD BE 1.19 VOLTS.
IN A WORKING CIRCUIT IT IS 19.9 VOLTS.

THAT FAULT DOES NOT EXPLAIN ANY OF THE SYMPTOMS YOU OBSERVED BUT IT IS
CONSISTENT WITH THE VALUES FOR
THE OUTPUT CURRENT OF THE CONSTANT CURRENT SOURCE, THE BASE EMITTER
VOLTAGE OF Q5 AND THE VOLTAGE AT N1.

>>SAVE
CONTEXT SAVED, TO RETURN TO IT TYPE 'RESTORE'.
YOU NOW HAVE A WORKING INSTRUMENT WITH WHICH TO EXPERIMENT.
```

The user has discovered more flaws in his understanding and is probably surprised by the high voltage between N4 and N5 that follows from his hypothesis (i.e. 31.3 volts). Wishing to explore what caused R22 to be blown and what lead to this unexpectedly high voltage that the current troubleshooting context be saved (i.e. all his measurements which are the gist for the evaluation specialist). He can now perform any experiments he wishes.

```
>>WHAT IS THE I THRU THE BASE OF Q3
THE BASE CURRENT OF Q3 IS 47 MILLIAMPS.

>>WHAT IS THE CURRENT THRU R22 WHEN IT IS SHORTED?
IF R22 WERE SHORTED
UNDER THE PRESENT SETTINGS, THE CURRENT THRU R22 WOULD BE 61 MILLIAMPS.
IN A WORKING CIRCUIT IT IS 1.8 MILLIAMPS.
```

This is an example of a hypothetical or conditional question.

```
>>WHAT ARE THE SPECS OF R22
THE SPECIFICATIONS OF R22 ARE:
VALUE -- 510 OHMS
POWER/RATING -- .5 WATTS
```

He now asks a factual questions which he could have just as well looked up.

```
>>RESTORE
PREVIOUS SITUATION RESTORED.
```

He now appears to understand why R22 blew when Q3 opened and wishes to get back to the task of troubleshooting.

```
>>WHAT COULD BE WRONG?
HERE ARE SOME FAULTS WHICH EXPLAIN THE MEASUREMENTS YOU OBSERVED:
THE BETA OF Q3 IS LOW
THE BETA OF Q4 IS LOW
OR R22 HAS THE WRONG RESISTANCE
```

He seems to have run out of ideas and requests some help. At this point the hypothesis generation "specialists" are invoked. These procedures examine all the measurements he has thus far made and then constructs a collection of hypotheses which are logically consistent with the values of information from these measurements.

```
>>WHAT WOULD THE OUTPUT VOLTAGE BE IF THE BETA OF Q3 WERE 8?
IF THE BETA OF Q3 WERE 8
UNDER THE PRESENT SETTINGS, THE OUTPUT VOLTAGE WOULD BE 10.5 VOLTS.
IN A WORKING CIRCUIT IT IS 19.9 VOLTS.
```

The user takes a wild guess probably to get a qualitative feel about how the beta of Q3 affects things.

```
>>IS THE EMITTER CURRENT OF Q4 CORRECT
THE EMITTER CURRENT OF Q4 IS -61 MILLIAMPS.
IN A WORKING CIRCUIT THE EMITTER CURRENT OF Q4 IS -49 MILLIAMPS.
THAT VALUE SEEMS TO BE INCORRECT.
```

This question is in some sense a cheat. It really consists of two measurements. If a cost were being assigned to his measurements or questions he would be charged accordingly.

```
>>COULD R22 BE LOW
IS IT OKAY IF I USE 25 OHMS FOR THE RESISTANCE OF R22?
> YES
THAT FAULT AGREES WITH ALL OF THE MEASUREMENTS YOU HAVE MADE.
```

Finally he generates a reasonable hypothesis, but this doesn't mean that R22 is necessarily low! Just that R22 being low is consistent with what he should have been able to determine from his measurements thus far.

```
>>WHAT IS THE RESISTANCE OF R22
THE MEASURED RESISTANCE OF R22 IS 25 OHMS.

>>REPLACE R22
R22 REPLACED
NOW YOU SHOULD TAKE MEASUREMENTS TO SEE IF THERE IS ANYTHING
ELSE WRONG.
```

The system decides he must know what is wrong with R22 so it doesn't hassle him like it did when he asked for Q6 to be replaced.

```
>>NOW WHAT IS THE OUTPUT
THE OUTPUT VOLTAGE IS 19.9 VOLTS.

>>WHAT ABOUT WITH A 10 OHM LOAD
WITH THE LOAD NOW SET TO 10
THE OUTPUT VOLTAGE IS 10 VOLTS.

>>WITH A 1000 OHMS
WITH THE LOAD NOW SET TO 1000
THE OUTPUT VOLTAGE IS 30 VOLTS.
```

Note the ability for the system to handle rather extensive ellipses where the context for "expanding" this phrase "with a 1000 ohms" is two questions back.

```
>>WHAT WOULD HAPPEN IF R22 WERE SHORTED?
IF R22 WERE SHORTED
UNDER THE PRESENT SETTINGS:
      CC=1.0, CR=HIGH, LOAD=10, VC=1.0, VR=HIGH
THERE WOULD BE NO CHANGE IN THE OUTPUT VOLTAGE (I.E. 30 VOLTS).

HOWEVER UNDER THE SETTINGS:
      CC=1.0, CR=HIGH, LOAD=10, VC=1.0, VR=LOW,
THE OUTPUT VOLTAGE WOULD BE .623 VOLTS.
IN A WORKING CIRCUIT IT IS 9.93 VOLTS.
```

A good example of what else is involved in handling hypothetical questions. Here the system first guesses that he is interested in output voltage since he didn't specify anything particular. Then it fails to find any change of the output voltage with the present load resistance. However, before it concludes that R22 has no effect it searches some other knowledge about the circuit and decides that a better test case or example would be to set the instrument up with a heavier load. Under this setting it does find a significant change.

```
>>IS ANYTHING ELSE WRONG?
NO
```

There are several points worth emphasizing about the dialogue. First, the student is not restricted to any particular order or number of measurements. This means that SOPHIE must be able to evaluate a student's hypothesis or generate a collection of theories about what is wrong relative to any possible set of measurements, control settings, loads, etc. that the student might decide to use. In other words, SOPHIE can't use pre-stored decision trees to help in any of these logical tasks and therefore must rely on powerful inferencing procedures. Second, the student in the dialogue is not a beginning electronics student. SOPHIE assumes that the user has the requisite electronic knowledge of someone beginning troubleshooting and is not prepared to answer such questions as "what is a transistor". Third although the hypothesis evaluation specialists refute his hypotheses quantitatively, the evaluation actually occurs at a qualitative level and only if inconsistencies are discovered which are the exact quantitative ramifications of his hypothesis presented to him. In the next section, we will discuss the mechanisms which allow SOPHIE to carry on dialogues.

Techniques for Inferencing

SOPHIE manifests most of its "intelligence" through its question answering and hypothesis evaluation and generation abilities. These abilities are achieved through a set of special purpose inferencing procedures each of which performs a certain class of inferences extremely efficiently. The centralizing component of the inferencing system is a simulation program modeling a "piece of knowledge" which in this case is an electronic instrument*. The underlying idea of how simulation can be used to perform inferencing is both straight-forward and extremely powerful. Let us first consider the problem of answering a hypothetical question (always with respect to a given circuit) of the form:

"If X then Y?"

where X is a proposition about some component in the given instrument and Y is a proposition about its behavior or symptoms. An example of such a hypothesis might be:

"If C2 is shorted, is the output voltage zero?"

The answer to the question can be found by invoking the simulator: First the simulation model of the instrument must be modified so that C2 is shorted (i.e., the proposition X must be made true on the model). Then the simulation of the modified model is executed. Since the

---

*More precisely, it models a schema of electronic instruments with one element of the schema being the working instrument and the other elements representing various ways the instrument can be faulted.

results of the simulation run contain all the consequences of the modification (C2 being shorted), the hypothetical consequent (the output voltage being zero) is simply checked against these simulation results.

The above paradigm skips over several logical difficulties concerning which boundary and/or input conditions should be used for the simulation runs. If it is necessary to determine all the logically possible consequences of a hypothetical modification, then the simulation must, in principle, be run over a potentially infinite collection of the instrument's control settings, etc. While for most practical situations there are only a finite number of cases "worth" considering, this number can still be quite large. It is clearly necessary to have an additional inferencing mechanism which can determine what the worthwhile cases are for any particular question. This additional mechanism must embody electronic knowledge of a different sort than is represented in the simulator. Thus, metaphorically, the simulator may be interpreted as creating examples whereas this additional mechanism tries to guarantee that these examples will be useful.

The tasks that fit most simply into the simulation paradigm concern requests for measurements. It is through this mechanism that SOPHIE can create the electronics laboratory within which the student is working. Whenever the student requests a measurement, the simulation is called to compute the voltage at every node in the circuit. From these voltages procedural specialists derive answers to additional questions about the current through any component, the resistance of a component, the power dissipation of a component, the beta of a transistor, etc. Whenever the student wishes to explore the circuit under different conditions, he can arbitrarily change the controls, modify any component, or introduce his own faults. Any such changes get efficiently translated by other procedural specialists into new simulation models.

Hypothesis Evaluation:

The first sophisticated use of simulation concerns the task of hypothesis evaluation. Remember that hypothesis evaluation requires a technique that can check the logical consistency of a hypothesis against the measurements the student has taken. For example, hypothesis evaluation is required when a student, after making several measurements, develops an idea (hypothesis) about what is wrong, e.g., "Is it possible that resistor R9 is open?".

To evaluate the given hypothesis we must derive all of its logical consequences and see if any of these consequences conflict with information derivable from his measurements. If there are such conflicts, they must be pointed out to the student as logical inconsistencies. In addition, evaluation should identify which of his measurements directly support his hypotheses and which are independent of it.

The evaluation strategy makes extensive use of simulation in the following way: First, the simulation model is modified so as to be consistent with the given hypothesis, i.e., the fault hypothesized by the student is inserted into a working model. Then all the student's measurements are repeated under this "hypothetical" model. For each measurement there are four cases that might occur. (1) The observed and hypothetical values may agree. (2) The observed value may represent a symptom (i.e., is wrong), while the hypothetical value is normal (i.e., is correct). In this case the fault proposed by the student does not account for this particular symptom. (3) The observed value may be normal while the hypothetical value is wrong. In this case the proposed fault would have created symptoms which the student did not observe. Or (4) the observed value and the hypothetical value may both be symptomatic but not the same. In every case but the first, the student must be told how the measurements disagree. The student's hypothesis is consistent if all of his measurements fall into case (1).

The comparisons needed to separate the above cases require knowing not only the values of a measurement in the hypothetical and malfunctioning circuits but also the value in a working circuit as well. The value in a working circuit is used to determine when the other two values differ qualitatively. For example, if the value that the student observed was 25 volts and the value under the hypothesized fault was 30 volts the difference between these two may or may not be qualitatively significant. If the value of the given measurement in the working circuit is 30 volts, the proposed fault does not account for the lower voltage observed in the faulted circuit. However, if the working circuit voltage is 3 volts, the hypothesis is doing a pretty good job of explaining the behavior implied by this measurement. Therefore, in addition to using simulation to determine the above quantities, a metric is involved to "judge" the qualitative distance between these values. The heuristic of using the metric to identify when two measurements significantly differ, provides a beautifully simple circumvention of the need for a "theory" of how and

why the instrument works. (See (Brown et al., 1974) for a complete specification of this process.)

The values of the student's measurements in a working circuit are also used to separate those measurements which support his hypothesis from those that are independent of it. If the values under all three conditions (i.e., the working circuit, the faulted circuit, and the hypothesis-related circuit) are essentially the same, the information derived from that particular measurement is independent of his hypothesis. If the faulted and hypothetical values agree but are different from the working value, the measurement supports his hypothesis.

Simulation Models

As we have seen, DC simulations form part of the basis of SOPHIE's understanding of electronics. We currently use two types of simulators. The first is a general purpose circuit simulation package coded in FORTRAN, called SPICE (Nagel 1971,1973). The other is a functional simulator written in LISP which incorporates circuit dependent knowledge.

There are many problems unique to our use of simulation. For example, methods of modeling a circuit which facilitate the insertion of faults had to be developed along with explicit models of faulty components. In addition, the faulting of one component will very often overload one or more other components leading to fault propagation. Such situations require a special monitoring mechanism which "sits on top" of the general purpose simulator and looks at the results of each simulation to decide if and how additional parts would blow. In fact, this mechanism, by making successive calls to the simulator, grows a fault propagation tree which captures the causal chain of events of several parts being blown by one initial fault. This tree also serves as a data base for some of the question answering routines.

CHAPTER 2 -- HYPOTHESIS GENERATION

Whenever the student requests help from SOPHIE, the hypothesis generation system is invoked. This system provides him with a list of possible faults which follow from or are logically consistent with the measurements he has thus far observed. The technique used is based on a brute force method which has been made more intelligent through the addition of several procedural specialists. We will first describe the brute force method and then describe how we have circumvented many of its limitations.

This brute force technique begins with a predetermined list of all the possible faults which may physically occur in the instrument. Then, for each measurement the student has taken, each fault on the list is run through the simulator. If the simulated value for each measurement under a particular fault agrees with the observed value, then this fault is consistent with what the student has seen. Otherwise this fault violates at least one measurement the student has taken and is therefore removed from the list of "viable faults."

This technique has several obvious limitations. The primary limitation is that of speed. With the large number of faults in the IP-28 and with a slow simulator like SPICE, the time required to simulate all physically possible faults would be excessive! A more theoretical limitation is that some components of the circuit have an infinite number of fault modes and hence might in principle require an unbounded number of simulation runs. Two examples of this are a faulted resistor whose value has changed or a faulted transistor whose beta has changed.

The most interesting way that the speed limitation is circumvented is by creating a specialist called the proposer. The proposer first looks at the result of one measurement and then uses that result to deduce a set of faults. This set of faults is smaller than the entire set of all possible faults for the instrument but still contains all the faults which are consistent with that one measurement. In addition to this technique we have constructed a special purpose functional simulator, (for the IP-28) which runs between 10 to 100 times faster than the general purpose simulator, but which is less accurate than the general purpose one.

The limitation encountered with those fault modes which have an infinite set of possible values is circumvented by a creating another specialist called the instantiator. For each component having such a fault mode, the instantiator uses the observed output voltage to determine the faulted value of the component. It obviously encodes a lot of special knowledge about the circuit to be able to carry out this task.

Both of these specialists work in conjunction with the third specialist called the refiner. This specialist uses the fast simulator (not SPICE) with routines which compare the observed faulted value with the value produced by the proposed fault. If these two values do not agree, then this fault is removed from the list of possible faults. It therefore refines the list of faults which have been proposed by the proposer.

Interaction of the Three Specialists

Let's take the following example. Assume that the student has taken only one measurement, the output voltage, and has found it to be low. At this point he asks for help. The proposer first examines the value observed for the output voltage and from that deduces a list of possible faults which could explain that measurement. It performs this task by using a set of procedures which encode sufficient circuit-dependent knowledge to be able to use only the output voltage and the settings, and from this information proposes a list of all the possible faults which are consistent with the observed output voltage. However, this list may also include some spurious faults which do not cause the observed behavior. The instantiator now goes to work. On those faults which require a value (an example in this case would be the beta of Q3 being low), the instantiator determines what this value should be. The refiner then runs each proposed fault through the fast simulator to confirm that the value of the output voltage that the simulator obtained is the same as that which the student observed. This refinement specialist thereby rules out those faults which were spurious or over-general.

The Proposer

The proposer is used to generate a set of probable faults. This specialist must propose every fault which can explain the behavior observed by the student. It also can, and does, propose faults which do not produce the behavior the student has seen. These are not excluded because the proposer does not take into account every nuance of the

circuit. To take every nuance into account, would require rules which are too complex and, in addition, these complex rules may not even be able to be determined. The rules used by the proposer are reasonably simple and fast. For example, when the output voltage is essentially zero volts there is one group of possible faults including Q3 being open, R16 being open and C5 being shorted. When the output voltage is 0.6 volts less than what it should be, the fault D6 being shorted is generated. There are approximately a dozen such rules for the IP-28 instrument.

The proposer uses only the value of the output voltage. Additional proposers could be written; however, one proposer would be required for each possible measurement in the circuit. Since good troubleshooting techniques require the student to take external measurements before internal measurements, our single proposer works quite well. Of course this means that the hypothesis generation facility cannot be used until an external measurement has been taken.

When the proposer is presented with an output voltage that is not symptomatic, i.e., the same as that in the working circuit, it can still generate possible faults. However, this list can be quite large. Certain faults act normally under most settings. For example, unless the load is sufficiently low, the fault of Q6 being open would not be detectable. Therefore, this fault cannot be ruled out until some symptomatic behavior is seen or until the settings have been changed such that a symptom should have been seen and wasn't.

The student could ask for the output voltage at several settings before asking for help. The proposer then makes a list of possible faults for each setting. Only faults which are on all the lists are then considered.

The Instantiator

There is a class of faults (suggested by the proposer) which have left unspecified a specific value for the fault. An example is for a transistor to have a changed beta. Here, the proposer claims the value of beta has changed but has not specified what the changed value is. However, before that fault can be simulated a specific value must be chosen. It is the job of the instantiator to determine such values.

The instantiator uses two techniques to determine the value for such faults. One technique is to directly calculate the value. An example is that of Q3 having

incorrect beta. The instantiator proceeds by dividing the output current by the amount of current coming out of the Constant Current Source to determine the current beta of the Darlington section. This is then divided by the beta of Q4 to determine the beta of Q3. This works because the values for the Constant Current Source and the beta of Q4 are assumed to be correct since only one component is faulted in the circuit.

The second technique involves approximation using "forward" functions. In certain cases one cannot determine the "inverse" function that translates the output voltage into the component value as was done for the beta of Q3 above. One can only determine the output voltage from the value of the component. The maximum and minimum values of the component are tried using the appropriate forward function. If the output value is not inside the range generated by the function, then it is not a possible fault. If it is inside the range the value at the midpoint is tried next. This binary search continues until the correct input value which causes the output value is found (within some percentage). This requires that the forward function being used is monotonic. In actuality, the forward functions only need to calculate the value at the outside of their functional block. Inverse functions are used to determine the functional block's value from the output value.

The above description assumes that an incorrect or symptomatic output voltage had been used. When dealing with a correct or non-symptomatic output voltage, less information about what the value should be is known. Only the upper bound on the value for the functional block can be determined. An example is that of Q3 having insufficient beta. With a sufficiently large load resistor, the output voltage will be correct. There will be a point, however, where the beta will be so low that incorrect output voltage would have resulted. Therefore, if the transistor Q3 is faulted in that manner and the output voltage is correct, the beta must be above that cut-off point.

If a component has been instantiated more than once, then it is possible that that fault should be removed from consideration. If both instantiations were based on symptomatic output voltages, then two exact values can result for the component. If these two values disagree then the fault is ruled out.

The second case is when the instantiations were based, one on a correct output voltage and the other on an errored one. If the exact value does not fall in the range produced

by the upper bound, then the fault is ruled out. If both instantiations were based on correct output voltages, then two upper bounds resulted. The lower for those two upper bounds is taken as the new upper bound.

The Refiner

The job of the refiner is to eliminate the spurious faults suggested by the proposer. The refiner is not limited to looking only at output voltages. It allows the comparison of any measurement that the fast simulator can generate. It takes a fault from those suggested by the proposer and a measurement (with a particular setting) that the student had previously taken and runs it through the fast simulator. It then compares the value from the simulator with the value the student observed. If the metric determines that the two values are not equivalent, then the fault is removed from further consideration.

The metric used for the comparison is the same as that used in hypothesis evaluation. It is described in detail in (Brown et al., 1974). This metric uses three values -- the value the student observed, the value produced by the simulator for the fault being explored, and the value in a working circuit. The tolerance used to compare the student's value and the simulator's value is proportional to the difference between these two values and the working circuit's value.

The value the student observed is re-determined by the fast simulator rather than SPICE. (The other two values were determined by the fast simulator previously.) This would be unnecessary except that the fast simulator is not as accurate as SPICE and a comparison between these three values is more meaningful when determined by the same simulator.

There is one case where the SPICE value must be used for the student's observed value. This is when the fast simulator cannot simulate the fault that is in the circuit. This occurs when there is a multiple component fault in the circuit. The hypothesis generation system, unable to generate multiple faults, will then generate those non-multiple faults which mimic the behavior the student observed. The SPICE simulator value must of necessity be used since the fast simulator cannot handle multiple faults.

The fast simulator described in Brown et al., 1974 only found the values at the outside of the functional blocks (e.g., the constant current source). This simulator has now

been augmented by adding a specialist for each functional block which can determine values internal to its functional block. When a measurement is taken inside a particular functional block, its specialist examines the result of the fast simulation and determines the values of the measurements internal to that functional block. Not all the specialists are invoked for each simulation run. Only those needed are invoked.

Measurement Verification

After a student makes a measurement, we would like to tell him whether or not it was a reasonable measurement, that is, whether it eliminates one or more possible faults.

Before the measurement is determined, the system internally calls the hypothesis generation system to determine the list of possible faults at that point in time. The measurement is then determined. Again the hypothesis generation system is called internally and a second list of possible faults is obtained which now takes into consideration all the old measurements plus this new one. These two lists are compared. If they are the same, then the student's measurement did not narrow down the list of possible faults and could then be considered unreasonable!

In the current implementation we type out the message that the measurement was useless. If there was only one fault remaining on the list of possibilities before he took the measurement, the student is told that he has enough information to uniquely determine the fault. The system then refrains from printing any more "useless measurement" messages to keep from further confusing the obviously confused student.

Another implementation being considered is to take a more positive approach and to print out a list of faults that have been removed from the possibility list after he takes a measurement. Unlike the first approach, that gave him only negative feedback, that is, that he did something wrong, this approach may prove less discouraging.

The information as to the faults that have been ruled out by his measurement is valuable to the student. If the first approach is used, we intend to allow the student to obtain this information by asking for it.

CHAPTER 3 -- A SEMANTICALLY CENTERED PARSING SYSTEM

When we started building SOPHIE's natural language processor we knew that it had to be both efficient and understanding of informal speech, i.e. "friendly". If either of these criterion were not met, letting the student use natural language would be an obstacle rather than an aid to the instruction process. If the natural language processor were not efficient, the student would lose interest while waiting for his question to parse. If it were not friendly, he would get frustrated trying to find a way to express his ideas.

While the problems involved in building an efficient system are well-known, those involved in a truly friendly system are not. We quickly discovered that when students use a system which exhibits "intelligence" in its deductive and inferencing capabilities, they start to assume that the system must also be intelligent in its conversational abilities as well. For example they would frequently delete parts of their statements which they felt would be obvious given the context of the preceding statements. This included the use of such linguistic phenomena as pronominalizations, anaphoric deletions and ellipsis. This led us to concentrate during the last six months on the semantic problems of dialogue.

Annotated Dialogue

Before we discuss how SOPHIE's natural language processor works, we present an annotated dialogue which is intended to demonstrate SOPHIE's new linguistic capabilities.* Even though the dialogue necessarily deals with electronic jargon, the linguistic issues it exemplifies occur in all domains. The annotations (lower case, indented) attempt to point out these problems and should be understandable to the non-electronics oriented reader. In the dialogue the student's typing is underlined.

---

*The reader is encouraged to see (Brown, 1974) for further examples and descriptions of SOPHIE's tutorial and inferential capabilities.

>>INSERT A FAULT

The student tells SOPHIE to give him a fault which he can troubleshoot. SOPHIE randomly selects a fault, inserts it into a model of the instrument and tells the student how the controls are presently set.

```
THE INSTRUMENT HAS A FAULT AND ITS PRESENT CONTROL SETTINGS ARE:
CC          1.0       (CURRENT CONTROL-FROM 0.0 (MIN CURRENT) TO 1.0)
CR          HIGH      (CURRENT RANGE SWITCH, HIGH=1 AMP, LOW=.1 AMP)
LOAD        1000      (LOAD RESISTANCE IN OHMS)
VC          1.0       (VOLTAGE CONTROL-FROM 0.0 (MIN VOLTAGE) TO 1.0)
VR          HIGH      (VOLTAGE RANGE SWITCH, HIGH=30 LOW=10 VOLTS)
```

>>WHAT IS THE OUTPT?
=OUTPUT
THE OUTPUT VOLTAGE IS 1.2 VOLTS.

SOPHIE corrects the student's typing mistake.
Notice that "the output" was clarified as being the output voltage.

IF YOU ARE NOT SURE WHETHER A VALUE IS CORRECT, YOU CAN ASK 'IS THAT VALUE CORRECT?'

"Correct" here means what it is in a working instrument.

>>IS THAT RIGHT
IN A WORKING CIRCUIT THE OUTPUT VOLTAGE IS 30.1 VOLTS.
THAT VALUE IS INCORRECT.

"That" here refers to the value of the previous measurement. SOPHIE, in addition to the faulted instrument, has a working circuit which can be used for reference.

>>WHAT IS THE VOLTAGE BETWEEN THE CURRENT LIMITING TRANSISTOR
...AND THE CONSTANT CURRENT SOURCE?
THE VOLTAGE AT N4 IS 2.48 VOLTS.

In addition to specific part names, SOPHIE recognizes references to parts by the their section. Conjunction is only handled in limited cases. Notice that even though the sentence says "voltage between x and y", it really means "voltage at z with respect to ground". The output routines reduce the input to its simpler form (i.e. between x and y reduces to N4).

>>WHAT IS IT BETWEEN THERE AND THE BASE OF Q6
THE VOLTAGE BETWEEN N4 AND N5 IS 1.28 VOLTS.

Here "it" refers to "the voltage" and "there" refers to N4. Also notice that "between" has to be treated differently, i.e. two nodes are explicit now.

>>Q5?
THE VOLTAGE BETWEEN N4 AND N10 IS 1.84 VOLTS.

An ellipsis. The student in effect meant "what is the voltage between N4 and the base of Q5?.

>>WHAT IS THE BASE CURRENT OF Q3
THE BASE CURRENT OF Q3 IS 12 MICROAMPS.

SOPHIE can answer questions about current, resistance, and power dissipation as well as voltage. The output and input routines know about unit prefixes.

>>THRU THE EMITTER?
THE EMITTER CURRENT OF Q3 IS -.26 MILLIAMPS.

Another ellipsis. Notice that the referents cannot be figured out on syntactic grounds and yet this is the sort of statement a student might make if the last answer had caused a slight shifting of his thinking.

>>COULD THE PROBLEM BE THAT Q5 IS BAD?
PLEASE BE MORE SPECIFIC ABOUT HOW Q5 IS FAULTED.

The student develops an idea about what is wrong and asks to have it evaluated.

>>COULD IT BE SHORTED?
THE SHORTED MODES OF A TRANSISTOR I HANDLE ARE: ALL JUNCTIONS SHORTED OR ONE JUNCTION SHORTED WITH THE OPPOSITE TERMINAL MODELLED OPEN.

A hypothesis must be completely specified (or non-ambiguous) before evaluation can take place.
Even though the last statement generated an error condition, it is still used to determine possible pronoun referents.

SHOULD Q5 HAVE FAULT MODE EB/EC/BC = SH/SH/SH?

```
> NO
WHICH TERMINAL SHOULD BE MODELLED OPEN?
> THE BASE
OK, I'LL USE EB/EC/BC = OP/SH/OP
```

At this point, SOPHIE looks back over the student's measurements to determine if he should have "known better". See (Brown, 1974) for further description.

```
HMMM, LET ME THINK ABOUT THAT FOR A MINUTE.
WHEN THE SETTINGS WERE:
        CC=1.0, CR=HIGH, LOAD=1000, VC=1.0, VR=HIGH,
THAT FAULT CONFLICTS WITH THE FOLLOWING MEASUREMENT.
YOU OBSERVED THAT THE VOLTAGE BETWEEN N4 AND N5 WAS 1.28 VOLTS.
IF Q5 HAD FAULT MODE EB/EC/BC = OP/SH/OP IT WOULD BE 1.1 MILLIVOLTS.
IN A WORKING CIRCUIT IT IS 1.43 VOLTS.
BUT THAT FAULT DOES EXPLAIN THE VALUES YOU OBSERVED FOR
THE OUTPUT VOLTAGE, THE VOLTAGE AT N4, THE VOLTAGE BETWEEN N4 AND N10,
THE EMITTER CURRENT OF Q3 AND THE BASE CURRENT OF Q3.


>>WHAT COULD BE WRONG
THERE ARE LOTS OF FAULTS WHICH WOULD EXPLAIN THE MEASUREMENTS YOU
HAVE TAKEN. HERE ARE SOME POSSIBILITIES:
THE RESISTANCE OF R8 CHANGES
THE LEAKAGE RESISTANCE OF C4 CHANGES
THE BETA OF Q3 CHANGES
R15 IS OPEN
R15 IS SHORTED
Q6 IS FAULTED
```

The student becomes stuck and asks for assistance. SOPHIE proposes some possible hypotheses for him to explore.

```
>>IF R8 WERE 30K, WHAT WOULD THE OUTPUT VOLTAGE BE?
IF THE VALUE OF R8 WERE 30000
UNDER THE PRESENT SETTINGS, THE OUTPUT VOLTAGE WOULD BE .904 VOLTS.
IN A WORKING CIRCUIT IT IS 30.1 VOLTS.
```

The student doesn't believe one of the possiblities so checks it using a hypothetical statement. Even though the voltage in this case and that in the faulted circuit are not the same, they qualitatively agree.

```
>>VOLTAGE DROP ACROSS R5
THE VOLTAGE ACROSS R5 IS -.028 MILLIVOLTS.
```

Noun phrase utterances are treated as questions.

```
>>WHAT ABOUT R3?
THE VOLTAGE ACROSS R3 IS 71.6 VOLTS.
```

Context of previous question is used to determine "what" about R3.

```
>>REPLACE R3
I AM GOING TO ASK YOU SOME QUESTIONS ABOUT HOW R3 IS FAULTED.
IS R3 OPEN OR SHORTED OR DOES IT HAVE THE WRONG VALUE?
> OPEN
R3 REPLACED

>>IS THERE ANYTHING ELSE WRONG
NO
```

After fixing the fault, the student uses SOPHIE to explore other aspects of the circuit's behavior.

```
>>IF C2 WERE LEAKY, WHAT WOULD THE VOLTAGE ACROSS IT BE?
IS IT OKAY IF I USE 10000 FOR THE LEAKAGE RESISTANCE OF C2?
> YES
IF THE LEAKAGE RESISTANCE OF C2 WERE 10000
UNDER THE PRESENT SETTINGS, THE VOLTAGE ACROSS C2 WOULD BE 26.3 VOLTS.
IN A WORKING CIRCUIT IT IS 28.9 VOLTS.

>>WHAT HAPPENS IF C1 SHORTS?
UNDER THE SETTINGS:
        CC=1.0, CR=HIGH, LOAD=1000, VC=1.0, VR=HIGH,
IF C1 WERE SHORTED, D3 WOULD BECOME OVERLOADED.
```

In the case where nothing more interesting happens, "what happens" questions default to the output voltage.

```
>>WHAT IS THE OUTPUT VOLTAGE WHEN THE VOLTAGE CONTROL IS SET TO .5?
WITH THE VOLTAGE CONTROL NOW SET AT .5
THE OUTPUT VOLTAGE IS 15.6 VOLTS.
```

A hypothetical question can involve the changing of a control.

```
>>IF IT IS SET TO .7?
WITH THE VOLTAGE CONTROL NOW SET AT .7
THE OUTPUT VOLTAGE IS 21.4 VOLTS.
```

Context determines which measurement should be taken and which

control should be set.

>>.9?
WITH THE VOLTAGE CONTROL NOW SET AT .9
THE OUTPUT VOLTAGE IS 27.2 VOLTS.

>>WHAT IS THE CURRENT THRU THE CC WHEN THE VC IS SET TO 1.0
WITH THE VOLTAGE CONTROL NOW SET AT 1.0
THE CURRENT THRU THE CURRENT CONTROL IS 30 MILLIAMPS.

CC and VC are acceptable abbreviations.

>>WHAT IS IT WITH IT SET TO .8?
WITH THE VOLTAGE CONTROL NOW SET AT .8
THE CURRENT THRU THE CURRENT CONTROL IS 24 MILLIAMPS.

Here it must correctly instantiate two occurrences of "it".

>>GOODBYE
GOODBYE

Use of Semantic Information During Parsing

In order to bring semantic knowledge* to bear on the problems of understanding a student's statement, the grammar from which the understanding program works is structured around semantic or conceptual categories rather than syntactic ones. This means that SOPHIE's parser, which operates in a top-down left-to-right manner, is working with conceptual entities. Before we discuss the actual mechanisms by which parsing occurs or the exact form of the "semantic" grammar, we will present some examples of how information is used during parsing.

Semantic Prediction in the Grammar:

One way that semantic information is used is to predict the possible alternatives that must be checked at a given point. Consider for example the operation of the parser when given the phrase "the voltage at xxx" (e.g., "the voltage at the junction of the current limiting section and the voltage reference source."). When it discovers the word

---

*Semantic information refers to all knowledge. No distinction is made between semantic, conceptual or pragmatic knowledge.

"at" in its top-down, left-to-right parse, it uses semantic information associated with the concept "measurement" to predict the nature of "xxx", i.e., it should be a noun phrase specifying a location (node) in the circuit. If "xxx" is not a location, the parser can give up without trying to treat "xxx" as any other possible noun phrase. This is similar to the effect of using semantic markers. However, a system using semantic markers would parse "xxx" as a noun phrase and then check for the marker +LOCATION. This would mean that marker checking could only be done after the work had been done to parse the noun phrase.

This same predictive information is also used to aid in the determination of referents for pronouns. If the above phrase were "the voltage at it", the grammar is able to restrict the class of possible referents to locations. By taking advantage of the available sentence context to restrict the class of possible referents, the simple rule of "last mentioned acceptable object" works in a large number of cases. Consider the sequence:

(1a) Set the voltage control to .8?
(1b) What is the current thru R9?
(1c) What is it with it set to .9?

In (1c), the grammar is able to recognize that the first "it" refers to a measurement that the student would like re-taken under slightly different conditions. The grammar can also determine that the second "it" refers either to a potentiometer or to the load resistance (i.e., one of those things which can be set). The referent for the first "it" is the measurement taken in (1b). The referent for the second "it" is "the voltage control" which is an instance of a potentiometer. The mechanism which selects the referents will be discussed later.

Using Local Information For Simple Omission:

Another capability of predictive semantic knowledge during parsing is the recognition of simple omissions. The parser knows, for each conceptual entity, the nature of its constituent concepts. When it is looking for an occurrence of a conceptual entity and cannot find an occurrence of a constituent concept, it can either fail (if the missing concept is considered to be obligatory in the surface structure) or hypothesize that a omission has occurred and continue.

For example, the concept of a TERMINAL has (as one of its realizations) the constituent concepts of a TERMINAL-TYPE followed by a PART. When the parser is

looking for the concept of a TERMINAL and only finds the phrase "the collector", it uses this information to posit that a part has been omitted (i.e., TERMINAL-TYPE gets instantiated to "the collector" but nothing gets instantiated to PART). In addition, it uses the dependencies between the constituent concepts to conclude that the omitted PART is a TRANSISTOR.

Since the parser is able to recognize things that have been left out, it can also sometimes fill in the missing piece. In the statement "change R9 to 1000", the parser notes that the units to be changed have been ommitted and decides that the student meant "ohms" instead of "watts".

Not all missing information is filled in by the local rules of the grammar. Given the question "What is the output?" and using the knowledge that a measurement needs a quantity and a place to measure it, the parser recognizes that some information has been omitted. Unlike the previous example, however, the missing information may be context-dependent. While, in most cases, the student means "What is the output voltage of the power supply?", if his previous question were "What is the input current of the Darlington Amplifier?", this interpretation is open to question. For this reason the decision as to the proper defaults is left to the procedural "specialist" in charge of calculating the answer to various kinds of measurements.

Recognition of Ellipses:

A third use for predictive semantic knowledge is to be able to accept elliptic utterances. These are utterances which do not express complete thoughts* but only give differences between the underlying thought and an earlier one. In this sequence, (2b) and (2c) are elliptic utterances.

(2a) What is the voltage at Node 5?
(2b) At Node 1?
(2c) What about between nodes 7 and 8?

The parser is aware of which constructs and which concepts are frequently used to contrast complete thoughts and recognizes occurrences of them as ellipses. While the parser is able usually to determine the intended concepts from the context available in a elliptic utterance, this is not always the case. Consider the following two sequences of statements.

---

*Note that the notion of a complete thought depends very much on the domain. For our purposes, a complete thought is a completely specified question or command.

(3a) What is the voltage at Node 5?
(3b) 10?
(4a) What is the output voltage if the load is 100?
(4b) 10?

In (3b), "10" refers to node 10, while (4b) refers to a load of 10. The problem this presents to the parser is that the concepts underlying these two elliptic utterances have nothing in common except the same surface realizations. The parser, which operates from conceptual entities, does not have a concept which includes both of these interpretations. One solution would be to have the parser find all possible parses (concepts) for a statement and then to choose between them on the basis of context. Unfortunately a great deal of time and effort is spent with such a method.

Capturing Semantic Information in the Parser

To enable the parser to use the semantic constraints, we have replaced the usual syntactic categories such as noun, noun phrase, verb phrase, etc., with semantic categories. These semantic categories represent conceptual entities known to the system, such as "measurements", "circuit elements", "transistors", "hypotheses", etc. While such refinement can lead to a phenomenal proliferation of non-terminal categories in a grammar, the actual number involved is directly related to the number of underlying concepts which can be discussed. For SOPHIE's present domain, there are approximately fifty such concepts.

The grammar which results from this refinement is a formal specification of the constraints that exist between the concepts that are of interest to the parsing process. The semantic grammar captures the ways of expressing a concept in terms of constituent concepts. Each rule also provides explicit information concerning which of its constituents concepts can be deleted or pronominalized. Part of the semantic grammar underlying SOPHIE's dialect abstracted as context free grammar is provided in Appendix A. Once the constraints have been formalized into the semantic grammar, the grammar is encoded as procedures. In this way, additional information peculiar to the recognition of a concept can be encoded in the corresponding rule (procedure). Writing a grammar as a program is not a new idea, the most notable prior example being Winograd's blocks world grammar (Winograd, 1973). The use of a procedural language allows complete freedom to embed arbitrarily complex information in the form of predicates and functions. Appendix B gives an example of the procedural specification of a grammar rule.

Representation of Information in the Grammar

There are two ways information is represented for use in the grammar. One way is procedurally in the grammar rules themselves. This is the way most of the predictive information is encoded. For example, the grammar rule which corresponds to the concept of measurement is written so that in parsing the phrase "the voltage at xxx", it will only try the grammar rule for location to parse "xxx".

The second way information is represented is as data in the semantic network which contains all of the time invariant data in SOPHIE. (See Brown, 1974 for a complete description of the semantic net.) For parsing, the semantic net is used to store information which links words to their possible concepts. For example, the information that Q4 is an instance of both the concepts of a TRANSISTOR and a PART is stored in the net.

Result of the Parsing

An advantage of basing the grammar on conceptual entities is that it eliminates the need for a separate semantic interpretation phase (i.e., the syntactic description stands in a one-to-one relationship with the semantic description). Since each of the non-terminal categories in the grammar is based on a semantic unit, each rule can determine the semantic description of the phrase that it recognizes in much the same way that a syntactic grammar determines a syntactic description. The low level rules build atomic "meanings" which get combined into functions by the higher level rules.

For example, the meaning of the phrase "Q5" is just Q5. The meaning of the phrase "the collector of Q5" is (COLLECTOR Q5) where COLLECTOR is a function encoding the meaning of "collector". "The voltage at the collector of it" becomes
(MEASURE VOLTAGE (COLLECTOR (PREF (TRANSISTOR))))
where MEASURE is the procedural "specialist" who knows about the concept of a "measurement". The "meaning" of the pronoun "it" phrase in the example is a call to the function PREF giving the possible classes of "it". PREF is also returned as the "meaning" of omitted phrases, i.e., there is no distinction made between something completely omitted and something which left "it" behind when omitted. The function PREF contains information about determining referents on the basis of context. The relationship between a phrase and its meaning can be straightforward and, if the concepts and target semantics are well matched, usually is. However it

can get complicated. In the phrase "the base of Q5 shorted to the emitter", the thing that is shorted is the base-emitter junction. Notice that the parser has some paraphrasing capabilities, as the "meaning" of the last phrase would be the same as "the base emitter shorted in Q5".

The top level "meaning" is a complete program (function call) which when evaluated calculates an answer to the student's question or performs the student's command. The program is also used by the output generation routines to construct the appropriate phrasing of the response to the student.

Determining Pronoun Referents

Once the parser has determined the existence of and the class (or set of classes) of a pronoun or omitted object, the context mechanism is invoked to determine the proper referent. This context mechanism has a history of student interactions during the current session which contains for each interaction the parse of the student's statement and the answer calculated by the system. To aid in the search of the "parses" on the history list,the context mechanism knows how each of the procedural specialists which can appear in a parse uses its arguments. For example, the specialist MEASURE has a first argument which must be a quantity and a second argument which must be a part, a junction, a section, a terminal or a node. When the context mechanism finds a match between a possible class of the pronoun and one of the argument positions of a specialist in a previous parse, the object in this argument position is checked. If this object is a member of one of the allowed pronoun classes, it is taken as the referent. The significance of checking the specialist during the search instead of just using the first object which satisfies the pronoun's requirements is that it avoids mis-interpretations due to object-concept ambiguity. For example, the object Q2 is both a part and a transistor. If the context mechanism is looking for a part, Q2 will be found only in those sentences in which it is used as a part and not in those in which it is used as a transistor. In this way the context mechanism finds the most recent occurrence of an object being used as a member of one of the desired classes.

Determining Elliptic Referents

If the problem of pronoun resolution is looked upon as finding a previously mentioned object for a currently specified use, the problem of ellipsis can be thought of as

finding a previously mentioned use for a currently specified object. For example:

(5a) What is the base current of Q4?
(5b) In Q5?

The given argument is "Q5" and the earlier function is "base current". This basic approach to ellipsis works surprisingly well. For a given elliptic phrase, the semantic grammar identifies the concept (or concepts) involved. In (5b), this would be TRANSISTOR. The context mechanism then searches the history list for a function in a previous parse which accepts the given class as an argument. When one is found, the new phrase is substituted into the proper argument position and the substituted meaning is used as the meaning of the ellipsis.

Fuzziness

Having the grammar centered around semantic categories allows the parser to be sloppy about the actual words it finds in the statement. The parser is willing to ignore some words trying to understand a statement. The amount of sloppiness (i.e. the number of words in a row which can be ignored) is controlled in two ways. First, whenever a grammar rule is invoked, the calling rule has the option of limiting the number of words that can be skipped. Second, each rule can decide which of its constituent pieces or words are required and how fuzzy the search for them can be. Taken together, these controls have the effect that the normal mode of operation of the parser is tight in the beginning of a sentence but more fuzzy after it has made sense out of something.

Prescanning

Before a statement is given to the parser, three operations are performed on the statement by a pre-processor. The first operation is the expansion of any abbreviations which occur in the statement. The second operation is a cursory spelling correction. The third operation is a reduction of compound words.

Spelling correction is attempted on any word of the input string which the system does not recognize. The spelling correction algorithm* takes the (possibly) misspelled word and a list of correctly spelled words and determines which (if any) of the correct words is close to

---

*SOPHIE uses the spelling correction routines developed by Teitelman for use in the DWIM facility of INTERLISP (Teitelman, 1974).

the misspelled word (using a metric determined by number of transpositions, doubled letters, dropped letters, etc.). During the prescan, the list of correct words is very small (approximately a dozen) and is limited to very commonly misspelled words and/or words which are critical to the understanding of a sentence. The list is kept small so that the time spent attempting spelling correction, prior to attempting a parse, is kept to a minimum. Remember that the parser has the ability to ignore words in the input string so we do not want to spend a lot of time correcting a word which won't be needed in understanding the statement. But notice that certain words can be critical to the correct understanding of a statement. For example, suppose that the phrase "the base emitter current of Q3" was incorrectly typed as "the bse emitter current of Q3". If "bse" were not recognized as being "base" the parser would ignore it and (mis-)understand the phrase as "the emitter current of Q3", a perfectly acceptable but much different concept.* Because of this problem, words like "base", are considered critical and their spelling is corrected before any parse is attempted. Note that there are a lot of words (e.g., "capacitor", "replace", "open", etc.) which if misspelled would prevent the parser from making sense of the statement but would not lead to any mis-understandings. These words are therefore not considered to be "critical" and would be corrected in the second attempt at spelling correction which is done after a statement fails to parse.

Compound words are single concepts which appear in the surface structure as a fixed series of more than one word. Their reduction is very important to the efficient operation of the parser. For example, in the question "what is the voltage range switch setting?", "voltage range switch" is rewritten as "VR". If not rewritten, "voltage" would be mistaken as the beginning of a measurement (as in "what is the voltage at N4") and an attempt would have to be made to parse "range switch setting" as a place to measure voltage. Of course after this failed, the correct parse can still be found but reducing compound words helps to avoid backtracking. In addition, reduction of compound words simplifies the grammar rules by allowing them to work with larger conceptual units. In this sense, the prescan can be viewed as a preliminary bottom-up parse that recognizes local, multi-word concepts.

---

*To minimize the consequences of such mis-interpretation, the system always responds with an answer which indicates what question it is answering, rather than just giving the numeric answer.

Conclusions

As stated in the beginning, our two goals for SOPHIE's natural language processor are efficiency and friendliness. In terms of efficiency the parser has succeeded admirably. The grammar is written in LISP which can be block compiled. Using this technique, the complete parser takes up about 5k of storage and parses a typical student statement consisting of 8 to 12 words in around 150 milliseconds! Appendix C provides exact timings for some of the statements in the dialogue.

Our goal of friendliness is much harder to measure since the only truly meaningful evaluation must be made when students begin using SOPHIE in the classroom. However, our results so far have been encouraging. The system has been used in hundreds of hours of tests by people involved in the SOPHIE project. In addition, about a dozen different people have had realistic sessions (as opposed to demonstrations) with SOPHIE and the parser was able to handle most of the questions which were asked. Anytime a statement is not accepted by the parser, it is saved on a disk file. This information is constantly used to update and extend the grammar.

The most problematic aspect of the natural language processor is its generality and extensibility. The approach taken to its development has been evolutionary: Add a new construct; see what other constructs it interferes with and what new statements it encourages students to use; and extend the parser to handle these. From the first sentence ever parsed, we were aware of the possibility that the next construct we wanted to add might be the "last straw" which forced us into a fundamentally different approach. However, we have not yet reached that limit. The results so far are sufficiently impressive that we feel it worthwhile to find out more about the limits of this technique. The most unnatural limitation in the grammar right now is the lack of conjunction. While incorporating conjunction will almost certainly require the addition of an interrupt mechanism similar to that allowed in PROGRAMMAR (Winograd, 1973), this is possible within the present framework. In fact, we believe that the semantic nature of our non-terminal categories and the predictive ability of the semantic grammar should provide a good handle on computational explosion normally accompanying conjunction. In any event, we believe that SOPHIE has demonstrated the feasibility of using natural language in mixed-initiative CAI systems.

REFERENCES

Brown, J.S. and Burton, R.R., "SOPHIE - A Pragmatic Use of AI in CAI", to appear in the Proceedings of the ACM Conference, November 1974.

Brown, J.S., Burton, R.R. and Bell, A.G., "SOPHIE: a Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An Example of AI in CAI)" BBN Report No. 2790, Bolt Beranek and Newman Inc. Cambridge, Mass. March 1974.

Brown, J.S., Burton, R.R., and Zdybel, F. "A Model-Driven Question Answering System for Mixed-Initiative Computer Assisted Instruction", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-3, May 1973.

Burton, R.R. "A Semantically Centered Parsing Strategy for Mixed-Initiative CAI Systems" paper presented at the 12 Annual Meeting of the Association for Computational Linguistics, Amherst, Mass., July 1974.

Carbonell, J.R. "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No. 4, Dec. 1970, pp. 190-202.

Carbonell, J.R. and Collins, A.M. "Natural Semantics in Artificial Intelligence," Proceedings of Third IJCAI, SRI Publications Department, Menlo Park, Calif. August 1973, pp. 344-351.

Goldberg, A. "Computer-Assisted Instruction: The Application of Theorem-Proving to Adaptive Response Analysis," Technical Report No. 203, Institute for Mathematical Studies in the Social Sciences, Stanford University, Stanford, California, May 1973.

Nagel, L.W. "Transient Analysis of Large Non-Linear Networks," IEEE Journal of Solid State Circuits, Vol. SC-6, August 1971.

Nagel, L.W. and Pederson, D.O. "SPICE: Simulation Program with Integrated Circuit Emphasis,"Memorandum ERL-M382, Electronics Research Laboratory, College of Engineering, University of California at Berkeley, April 1973.

Sleeman, D.H. "A Problem Solving Monitor for a Deductive Reasoning Task," International Journal of Man-Machine

Studies. (In press.)

Teitelman, W., INTERLISP Reference Manual, XEROX PARC, Palo Alto, Calif., February 1974.

Winograd, T., Understanding Natural Language, Academic Press, New York, 1973.

Woods, W.A., "An Experimental Parsing System for Transition Network Grammars," Rustin, ed. Natural Language Processing, Algorithmics Press, Inc., New York, 1973.

APPENDIX A

Formal Description of Part of the Grammar

This appendix gives a formal description of the language accepted by SOPHIE. The grammar is implemented as LISP functions and some examples are listed in Appendix C. The parsing process is sketched out and a list of compound words and abbreviations are given.

In the description, alternatives on the right-hand side are separated by ! or are listed on separate lines. Brackets [] enclose optional elements. An asterisk * is used to mark notes about a particular rule. Non-terminals are designated by names enclosed in angle brackets <>.

```
<circuit/place>:= <terminal> ! <node>

<diode/spec> := <diode> ! <zener/diode>
                <section> diode ! <section> zener/diode

<junction> := <junction/type> [of] <transistor/spec>
              <transistor/term/type> and <transistor/term/type> [of]
                   [<transistor/spec>]
              <transistor/term/type> to <transistor/term/type> [of]
                   [<transistor/spec>]

<junction/type> := eb ! be ! ec ! ce ! cb ! bc

<meas/quant> := voltage ! current ! resistance* ! power
                 *means measured resistance

<measurement> := <section>[output*][<meas/quant>]
                 output*<meas/quant>[of]<section>
                 output* [<meas/quant>] [of <transformer>]
                 <transformer> <meas/quant>
                 <meas/quant> between** <circuit/place> and*
                   <circuit/place>
                 <meas/quant> of*** <part/spec>
                 <meas/quant> between output terminals
                 <meas/quant> of <junction>
                 <meas/quant> of <circuit/place>
                 <meas/quant> from <junction>
                 <meas/quant> of <section>
                 <meas/quant> of <pronoun>
                 <junction/type> <meas/quant> [of <transistor/spec>]
                 <transistor/term/type> <meas/quant> of
                   [<transistor/spec>]
                 *input also
```

```
                **from-to also works
                ***at, thru, in, into, across and through also work

<node> := junction of <part/spec> and <part/spec>
          node between <section> and <section>
          [point] between <part/spec> and <part/spec>
          <node/name> ! [node] <node/number>
          <pronoun>

<part/spec> := <part/name> ! <load/spec> ! <section> <part/type>
               <pronoun>

<pot/spec> := cc ! vc ! cct

<pronoun>:= it ! [that] "type"

<terminal> := output [terminal] ! <transistor/term> ! center/tap
              positive terminal [<part/spec>] ! positive one
              negative terminal [<part/spec>] ! negative one
              anode [<diode/spec>] ! cathode [<diode/spec>]
              wiper [<pot/spec>]

<transistor/spec> := <transistor> ! <section> transistor ! <pronoun>

<transistor/term> := <transistor/term/type> [<transistor/spec>]

<transistor/term/type> := base ! collector ! emitter
```

<transistor>, <capacitor>, <diode>, <resistor>, <transformer> and <zener/diode> all check the semantic network and parse correct part names, e.g., R9, Q6.

<section> uses the semantic network to determine if a word is a section of the unit, e.g., current/limiter.

<part/name> uses the semantic network to see if a word is the name of a part e.g., R6, C4, T2.

<node/name> checks semantic network for node names.

APPENDIX B

A Rule from the Grammar

The grammar is encoded as LISP procedures. The ways of expressing a non-terminal are embodied in a grammar function. Each grammar function takes at least two arguments; STR, the list of words to be recognized, and N, the degree of fuzziness allowed. This function, in effect, must determine whether the beginning of the string STR contains an occurrence of the corresponding non-terminal. There are generally two types of checks that a grammar function performs. One is a check for the occurrence of a word or words which satisfies certain predicates. This checking is done with two functions -- CHECKLST and CHECKSTR. CHECKLST looks for a word in the string matching any of a list of words. CHECKSTR looks for a word in the string satisfying an arbitrary predicate. It is through these functions that the parser implements its fuzziness. For example, if CHECKSTR is called with the string "resistor R9" and a predicate which determines if a word is the name of a part (e.g., "R9"), CHECKSTR will succeed by skipping the word "resistor", which in this phrase is a noise word.

The other usual type of operation performed by the grammar functions is to check for the occurrence of other non-terminals. This is done by calling the proper function (grammar rule) and passing it the correct position in the input string.

If a grammar rule is successful, the function passes back two pieces of information. First, it returns some indication of how much of the input string has been accepted (i.e., where it stopped). The convention adopted is that the grammar rule returns as its value a pointer to the last word in the string accepted by the rule. Second, the function passes back a structural description of the phrase that was parsed. This structure is passed back in the free variable RESULT (analogous to an ATN's "*" upon return from a PUSH (Woods, 1973)).

Listed below is the grammar rule for the concept of a junction of a transistor. This rule accepts phrases such as "base emitter junction of Q5", "BE of the current limiting transistor", or "collector emitter junction".

```
(<JUNCTION>
  [LAMBDA (STR N)
    (PROG (TS1 R1)
          (RETURN
            (AND
(* comment A)
                [OR (AND (SETQ TS1 (<JUNCTION/TYPE> STR N))
                         (SETQ R1 RESULT))
                    (AND (SETQ TS1 (<TRANSISTOR/TERM/TYPE> STR N))
                         (SETQ R1 RESULT)
                         [SETQ TS1
                           (<TRANSISTOR/TERM/TYPE>
                             (CDR (CHECKLST (CDR TS1)
                                            (QUOTE (AND TO]
                         (SETQ R1 (JUNCTION-OF-TERMS R1 RESULT]
(* comment B)
                (COND
                  ([SETQ STR (<TRANSISTOR/SPEC>
                       (CDR (GOBBLE (GOBBLE TS1 (QUOTE (JUNCTION)))
                                    (QUOTE (OF))
                                    1]
                   (SETQ RESULT (LIST R1 RESULT))
                   STR)
                  ([SETQ RESULT (LIST R1 (LIST (QUOTE PREF)
                                               (QUOTE (TRANSISTOR]
                    TS1]))
```

Comment A:

The first thing that is looked for is either a <junction/type> (BE, emitter collector, etc.) or two <transistor/terminal/type>s (base, emitter or collector) separated by the words "and" or "to". If two terminals are found, the function JUNCTION-OF-TERMS is called to determine the correct junction. In either case, the place where the successful subsidiary rule left off is saved in TS1 and the meaning of the accepted phrase is saved in R1.

Comment B:

The next thing a junction needs is a transistor (<TRANSISTOR/SPEC>). <TRANSISTOR/SPEC> looks for an occurrence of a transistor e.g. "Q5" or "current limiting transistor". GOBBLE is a function for skipping relational words when they are not used to restrict the remaining part of the phrase. If a transistor is not found, a deletion is hypothesized and a call to PREF is constructed. If the

transistor has been pronominalized as in "the base emitter of it", <TRANSISTOR/SPEC> would recognize "it". In either case the semantics of the recognized phrase (something like (EB Q5)) is put into RESULT and a pointer to the last recognized word is returned as the value of <JUNCTION>.

At present there are approximately 80 grammar rules in SOPHIE's grammar.

APPENDIX C

Sample Parses and Parse Times

The following are examples of statements handled by the natural language processor. Under each statement, the semantic interpretation returned by the parser is given. The semantic interpretation is a function call which when evaluated performs the processing required by the statement. Parse times are given in milliseconds.

Insert a fault.
(INSERTFAULT NIL) (85 ms)

What is the output voltage
(MEASURE VOLTAGE NIL OUTPUT) (80ms)

What is the voltage between the current limiting transistor and the constant current source.
(MEASURE VOLTAGE (NODE/BETWEEN (FINDPART CURRENT/LIMITER TRANSISTOR)
CURRENT/SOURCE)) (335 ms)

What is the voltage between there and the base of Q6?
(MEASURE VOLTAGE (PREF (NODE TERMINAL)) (BASE Q6)) (100 ms)

Q5?
(REFERENCE ((TRANSISTOR) Q5)) (95 ms)

Could the problem be that Q5 is bad?
(TESTFAULT Q5 BAD) (100 ms)

Could it be shorted?
(TESTFAULT (PREF (PART JUNCTION TERMINAL)) SHORT) (75 ms)

If R8 were 30k what would the output voltage be?
(IFTHEN (R8 30000.0 VALUE) (MEASURE VOLTAGE NIL OUTPUT)) (220 ms)

If C2 were leaky what would the voltage across it be? (120 ms)
(IFTHEN (C2 LEAKY) (MEASURE VOLTAGE (PREF (PART JUNCTION)) NIL))

What is the output voltage when the voltage control is set to .5
(RESETCONTROL (STQ VC .5) (MEASURE VOLTAGE NIL OUTPUT)) (170 ms)

What is it with it set at .6?
(RESET CONTROL (STQ (PREF (POT LOAD SWITCH)) .6) (REFERENCE NIL))
(110 ms)

If it is set to .9?
(RESETCONTROL (STQ (PREF (POT LOAD SWITCH)) .9) (REFERENCE NIL))
     (135 ms)

What is the current thru the CC when the VC is set to 1.0
(RESETCONTROL (STQ VC 1.0) MEASURE CURRENT CC NIL)) (190 ms)